Analysis of human activities and identification of uncertain situations in context-aware systems

Leandro O. Freitas[1], Pedro Rangel Henriques[1] and Paulo Novais[1]
[1]*ALGORITMI Centre, Department of Informatics, University of Minho, 4710-057, Campus de Gualtar, Braga, Portugal*
leanfrts@gmail.com, {prh, pjon}@di.uminho.pt

Keywords: Context-Aware Systems, attribute grammar, quality of context, uncertainty identification.

Abstract: One of the main obstacles faced by context-aware systems developed for human activities recognition is related to dealing with incomplete data for decision making. Problems from several sources contribute to this phenomenon. Imprecise acquisition of data from sensors and the system's design issues are among them. This paper presents an approach to tackle this trouble by dividing it into three parts. The first one refers to the validation of context data through an Attribute Grammar. The formalism and expressiveness provided a grammar, enriched with attribute evaluation rules and contextual constraints, ensures that the system will use only valid data for reasoning. Anomalous data will be detected, and the situation will be signalized. Also, the analysis of Quality of Context is provided, considering a set of characteristics, vouching that only useful information will be considered. At last, the identification of the sources of uncertain situations followed by a sequence of actions aiming to minimize the negative impacts of it helps the system to work with more complete sets of data. The formalization of the approach is provided together with an algorithm to validate it.
CCS Concepts: • **Human-centered computing** → **Ambient intelligence**; Ubiquitous and mobile computing systems and tools; Ubiquitous and mobile computing design and evaluation methods; • **Theory of computation** → **Grammars and context-free languages**; Program semantics; Program analysis.

1 Introduction

Context awareness is in evidence nowadays. Personalization of applications and services, and the improvement of personal virtual assistants, are among the factors that contribute to this phenomenon. Context-aware systems developed for residences are seen as a fertile domain with a wide range of applications, for instance, Activity Daily Living (ADL). Independent of the system's goals, it must use a real-world representation environment as a basis. This allows the system to adapt itself according to different situations and assist users. For the scope of this paper, the domain of study refers to an intelligent environment, more specifically smart houses that monitor human activities.

Context-aware systems use the information provided by sensors that monitor and environment and creates formal representations that reflect what happens in the real world. This includes, for instance, the current states of entities such as users (local, actions, and emotional and physical states) and devices/appliances used by them. In this manner, according to (Freitas et al., 2018), context can be defined as anything, external to the users, that could impact their behavior or state of mind.

Several approaches in the literature address computational representations of these intelligent environments. All of them state that a formal and strict representation must be used to avoid ambiguous or imprecise interpretations. Also, they must be expressive enough to map variations of similar real-world situations.

In this paper, an Attribute Grammar-based representation of human activities for residences is presented. Attribute Grammar (AG) (Knuth, 1968) is a well-known field of investigation with several contributions to the area of programming languages. The main advantage of using this approach is the possibility of lexical, syntactic, and semantic validation of human activities due to its strict formalism. Along with this representation, it is possible to perform semantic analysis through evaluation rules, aiming for the improvement of the system's knowledge.

Even when using a well-defined representation of the domain in question, some critical issues arise and must be tackled to ensure the accuracy of the decision making of the system. The problem of uncertainty

is also addressed in this paper. It is generated when context-aware systems use low-quality context data for reasoning. Outdated, wrong, or absence of data might be sent to the system when sensors are not operating as they should. Each of these problems leads the system to work with different levels of uncertain information (White et al., 2018). The importance of this topic is in the fact that by using imperfect data, a context-aware system would create wrong interpretations of situations (Anderson et al., 2015).

The identification of causes that makes the system work with low-quality data helps to prevent problems generated by it. The analysis of the Quality of Context (QoC) (Buchholz and Schiffers, 2003) helps the system to identify problems related to data provided by sensors and decreases the frequency of undesired situations.

Considering that, this paper presents an approach for the improvement of the accuracy of context-aware applications. The main goal is to describe an attribute grammar that validates ordinary daily activities in a smart house and the definition of semantic rules to improve the capacity of the system to interpret context changes. Allied with this, it presents an approach to analyze the QoC. In this aspect, the objective is to create means to ensure that the system considers only useful data for decision making or, at least, to help it to identify problems in the inputs. The validation was conducted through the definition of an algorithm containing functions with a series of actions to identify and minimize uncertainty.

The paper is structured as follows: related work is described in section 2. Section 3, describes concepts of AGs and how they can be applied to intelligent environments since not much is known about their relationship. It presents an example of AG for human activity recognition aiming to demonstrate its usefulness. Characteristics of uncertainty are presented in section 4, where the importance of the quality of context information is discussed as well as the advantages of identification of uncertain scenarios for minimizing the impacts of it in human activities recognition. The validation of the proposal is described in detail in section 7. At last, the discussion of the results and final considerations of the paper are presented in section 7.

2 Related Work

One of the main contributions of this work is the description of an attribute grammar applied to the intelligent environment. At least to the best of the authors' knowledge, there are no similar approaches connecting these two fields of research. There are works in the literature with relevant contributions that can be related to the one presented here, even though not using attribute grammars to model human activities. Most of them apply decomposition of tasks into subtasks and actions, creating a hierarchy, and considering temporal aspects.

In (Giese et al., 2008), the authors state that formal modeling of tasks helps users to perform activities with computational support. For instance, it is possible to define rankings of priority of activities to be performed according to the user's context, considering, for example, safety and social skills. The authors consider three parameters: simulation of time and conditions, referring to preconditions for tasks and possible obstacles; spatial behavior, that analyzes the location where the tasks will be performed, the objects used and the possibility of changing actors from one location to another, and; analyzing communication, referring to the hierarchy and dependence.

In (Bolton et al., 2011), the authors describe the syntax and semantics of a function model capable of integrating task analysis with formal verification called EOFM (Enhanced Operation Function Model), based on Extensible Mark-up Language (XML). The model standardizes changes related to activities, and it defines the behavior of them according to their goals, based on the analysis of state and derived variables.

In (Mori et al., 2002) is presented the ConcurTaskTree Environment (CTTE), in which the goal is to provide means for the management of task models for cooperative applications. Among the features, it is interesting to highlight the precise definition of temporal logics for task and their relations; the creation of categories such as user tasks (human routine activities), application tasks (related to the system), interaction tasks (relating user and the system) and abstract tasks (tasks belonging to more than one category) (Mori et al., 2002).

In (D'Aniello et al., 2017) a situation awareness framework is proposed. The main goal is to use an approach for adaptive goal selection, where users can choose actions to take aiming to achieve goals, based on their desire, but also considering the cohesive options. The framework is integrated into an intelligent environment, and context data acquired by sensors feed it. It uses reinforcement learning to understand and improve the reward of the suggested options for the achievement of goals. The proposal is validated through a case study with a prototypical system.

Uncertainty evidenced in activity recognition is tackled in (Noor et al., 2016) through an approach combining ontology and Dempster-Shafer theory. The authors state that ontology-based solutions are limited due to problems related to the observations of

human actions. By quantifying this and taking into account additional context information, it is possible to define belief states, in order to improve the accuracy of the recognition process. Degrees of belief are also used in Speculative Computation (Satoh, 2005) to fill gaps of context data that generate uncertain scenarios. As well as the approach presented here in this paper, the authors of (Noor et al., 2016) consider an activity as being a sequence of actions. However, here activities are represented through attribute grammars instead of ontology.

3 Attribute Grammars applied to Intelligent Environments

Attribute Grammar (AG) is a wide field of investigation with several relevant contributions regarding the validation of programming languages. It introduces semantic rules into Context-Free Grammars, inserting meaning to the analysis of the grammar (Knuth, 1968). Still, according to (Knuth, 1968), semantics for the rules is achieved through the definition of attributes for non-terminal symbols by associating them with each production. This way, they are defined to manipulate attributes, which are associated with the grammar symbols.

The lexical and syntactical phases of processing precede the semantic. The lexical analysis has as primary goal to analyze the inputs and convert them into a set of terminal symbols. The result of this processing is used in the syntactical phase to generate an Abstract Syntax Tree (AST) (Bürger et al., 2010). These two first phases can be applied to intelligent environments once this kind of domain should be filled with sensors sending raw data to be analyzed by a context-aware system. Then, the semantic analysis is performed to take unevaluated attributes as input and return evaluated ones.

Another characteristic of AGs is the possibility of manipulation of the attribute values at any node of the tree (Bürger et al., 2010). This feature is essential once in an intelligent environment the states of entities (attribute values in AGs) may change frequently, and their values must be available to be used by the system at any time, with different purposes. A generic Abstract Syntax Tree of an intelligent environment is presented in figure 1.

The terminal symbols, in **a**, represent the raw data acquired from sensors, in **b**. It refers to any context information from the environment (e.g., humidity, level of lightness, localization, the presence of entities in a room). It will be used to characterize situations. In the general structure **a**, it will be characterized as been synthesized or inherited attributes and can be analyzed in different nodes of the tree. The entities represent the non-terminal symbols in an AG and may assume all the characteristics of them. They can be composed of several data properties (attributes). Besides that, attribute values from one entity may be used by other entities. In these cases, they will assume synthesized or inherited features. From the generic AST, it is possible to define rules to validate the structure that will be applied context-aware system.

Thus, it is possible to state that AGs can contribute to the consolidation of intelligent environments through different approaches. In this paper, it will be addressed the possibility of applying AGs for the validation of the activities of users.

3.1 Grammar for human activity recognition

When using AG, the validation of sets of inputs is performed through the execution of production rules. They specify the formalism that ensures that each entry passes through a verification regarding its composition. Considering human activities in smart houses, it was defined that it is essential to monitor the sensors that collect data, timestamps of the activities, local where the user performs them, devices, appliances or objects, and a description with additional information. With this data, the following production rules were created, aiming at the validation of daily human activities in a smart house.

p1: $context \rightarrow activity+$
p2: $activity \rightarrow id, record$
p3: $record \rightarrow date\_begin, hour\_begin, date\_end, hour\_end, local, description*$
p4: $local \rightarrow place, room$
p5: $date\_begin \rightarrow sensor\_id, DATE$
p6: $hour\_begin \rightarrow sensor\_id, HOUR$
p7: $date\_end \rightarrow sensor\_id, DATE$
p8: $hour\_end \rightarrow sensor\_id, HOUR$
p9: $place \rightarrow sensor\_id, appliance$
p10: $room \rightarrow TEXT$
p11: $description \rightarrow TEXT$
p12: $sensor\_id \rightarrow TEXT$
p13: $appliance \rightarrow TEXT$

The first rule (**p1**) defines the axiom of the grammar. It means that it restrains the type of acceptable inputs. The verification performed in all other production rules will affect this axiom. Considering the domain of daily human activities, it states that a context can be composed of one or more activities. This was defined because, depending on what the user performs, it will be necessary to integrate two or more small actions to represent an activity and, consequently, the context of the user. The second rule (**p2**) states that each activity performed by the user must contain an identification code of the sensor that sent the data

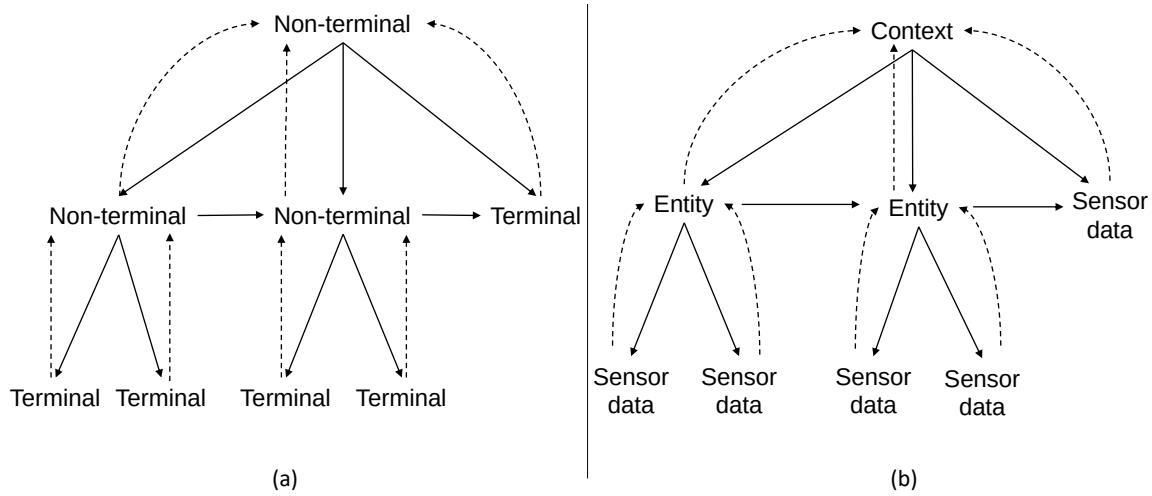

Figure 1 **a)** General structure of AST. **b)** Generic AST for intelligent environments.

and a record. Identification of sensors is important once, one of the phases when dealing with uncertain situations is to identify the source of it. Thus, the system should perform a sensor's diagnostics to find problems related to a specific one. More details about this process is given in section 6.

Rule **p3** defines the structure of record. In this case, it states that they should be composed of timestamps that identify the beginning and ending date and hour of the occurrences. Depending on how the system was developed and the types of sensors available, they can provide additional data. Thus, this rule also has a *description* symbol to ensure that any extra information about an activity will be considered (**p11**).

Rules **p5** and **p7** verifies the structure of the dates of the beginning and ending that composes the timestamps. Rules **p6** and **p8** check the beginning and ending hours of it. Rule **p4** defines the structure of the place and room of the house where an activity is being performed. The *place* symbol in **p9** validates the specific local, furniture or appliance where sensors are installed (**p12**). It also validates its identification code. The symbol *room* validates the local of the house where the user is executing the tasks (**p10**).

Following, figure 2 presents an AST of an attribute grammar applied to a case study described in (Lyons et al., 2010). The case study was adapted for the scope of this project. It describes a scenario where a woman is sleeping at her bed, as she usually does. However, in the middle of the night, at 1:00 am, she wakes up feeling bad, with a stomach ache. The user decides to take medicine to relieve the pain. Motion and door sensors provide data for the system to infer that the user is going to the medicine cabinet in the bathroom. The woman takes the proper medicine and returns to bed. This activity has preparation steps (leaving the bed and going to the bathroom) and the action itself of having the medicine.

Aiming a better visualization, the AST was divided in two parts **a** and **b**. The former contains data related to the preparation steps. In this case, it represents a scenario where the user wakes up during the night, leaves the bedroom, and goes to the local where the medicines are stored. The latter contains data related to the action of taking medicine. Together, **a** and **b**, are children of the root of the tree, represented by the *context* symbol.

The values of the attributes in this example refer to data acquired from sensors. After adequately analyzing this raw data, the system fills the structure that will be used by the grammar to validate the activities. Thus, the attributes of the symbols in the grammar are represented by context data.

The preparation steps that the user accomplish before performing an activity itself are naturally necessary (in the example, represented by the process of leaving the bedroom). Thus, it is crucial for the system to understand them. Considering the structure of the grammar and the values of attributes for the activity of *takeMedicine*, the productions will have the following behavior:

p5: $date\_begin \rightarrow dt001, 2019-01-21$
p6: $hour\_begin \rightarrow h001, 1:00$
p6: $date\_end \rightarrow dt001, 2019-01-21$
p7: $hour\_end \rightarrow h001, 1:01$
p7: $local \rightarrow door, bedroom$
p7: $description \rightarrow leave\_bedroom$
p5: $date\_begin \rightarrow dt003, 2019-01-21$
p6: $hour\_begin \rightarrow h002, 1:02$
p6: $date\_end \rightarrow dt003, 2019-01-21$
p7: $hour\_end \rightarrow h002, 1:05$
p7: $local \rightarrow medicine\_cabinet, bathroom$
p7: $description \rightarrow take\_medicine$

The system should have a module dedicated to the identification and analysis of patterns of behavior to facilitate context interpretation. By understanding the steps that the user usually performs to accomplish an activity, it is possible to identify the data that should be considered to model new activities. For instance, based on past occurrences, the system identifies that occasionally when the user wakes up during the night, she goes to the bathroom, open the medicine cabinet, and take medicine. Thus, it is possible to infer that all these steps are part of the same activity. The AG will use this data as synthesized and inherited attributes for the symbols to validate the structure of the activity.

The validation is done through the analysis of each of the steps that composed them, considering proper parameters. Besides that, the system identifies the conclusion of

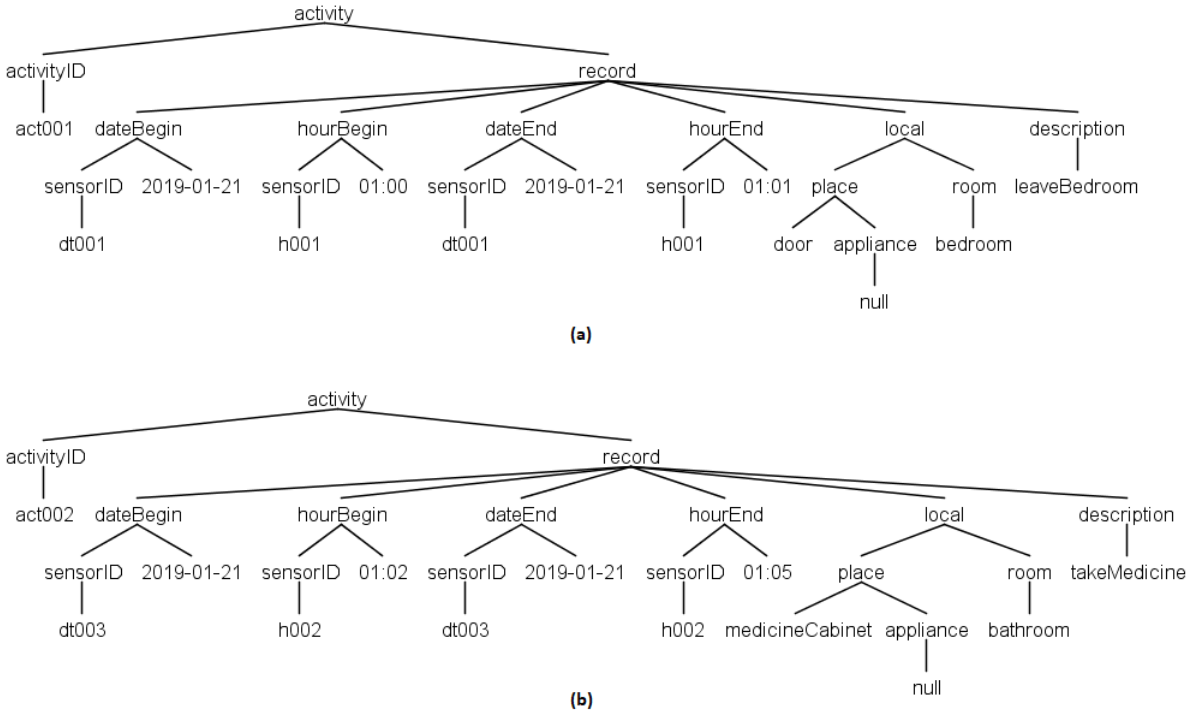

Figure 2 Abstract Syntax Tree: (a) leave the bedroom. (b) take the medicine.

activity by analyzing the behavior of the user. It considers the time used to finish the actions, the record of the activity, and any other relevant information. If the values of these attributes are within a predefined acceptable range, the system can infer that the activity as concluded with success.

The mapping and monitoring of activities are essential because any changes identified during the execution of an activity may be related to problems with the user, for example, if considering an Ambient Assisted Living domain, the unexpected changing of behavior can characterize health problems.

3.2 Semantic rules

AGs allows the definition of inference rules that can be used to manipulate context data properly. The execution of these rules increases the system's knowledge once the amount of available data for reasoning will be more significant. The following list presents some examples of semantic rules developed for the validation/ identification of daily human activities to be used by context-aware systems.

1. record.duration = getActivityDuration (activity.activityID, record.date_begin, record.date_end, record.hour_begin, record.hour_end);
2. activity.activityID = getActivityID(TEXT.value);
3. record.date_begin = getDate(DATE.value);
4. record.date_end = getDate(DATE.value);
5. record.hour_begin = getDate(HOUR.value);
6. record.hour_end = getDate(HOUR.value);
7. activity.identification = activityDescovery (activity.activityID, place.sensor_ID[], place.appliance[], local.room);
8. place.sensor_ID = getSensor(TEXT.value);
9. place.appliance = getAppliance(TEXT.value);
10. local.room = getRoom(TEXT.value);

Rule 1 calculates the time spent by the user to perform an activity. To accomplish that, it analyses the beginning and ending dates and hours. If the ending date and time are not available, it can use the current system's timestamps to finish the processing. To perform this calculation it is necessary to use the results from the execution of rules 2, 3, 4, 5 and 6. If the total is out of a pre-defined range of an acceptable amount of time, it may represent that the user is having problems to accomplish the task. Considering the case study of taking medicine during the night, the structure of the semantic rules would be the following:

1. record.duration = getActivityDuration (activity.activityID, record.date_begin, record.date_end, record.hour_begin, record.hour_end);
2. activity.activityID = act001;
3. record.date_begin = 2019-01-21;
4. record.date_end = 2019-01-21;
5. record.hour_begin = 01:00;
6. record.hour_end = 01:01;
1. record.duration = getActivityDuration (activity.activityID, record.date_begin, record.date_end, record.hour_begin, record.hour_end);

2. activity.activityID = act002;
3. record.date_begin = 2019-01-21;
4. record.date_end = 2019-01-21;
5. record.hour_begin = 01:02;
6. record.hour_end = 01:05;

It is possible to evidence that to calculate the time of the activity of taking medicine is necessary to execute rules 2 to 6 twice, changing only the parameters. This occurs because the activity is composed of two sets of actions. One is referring to waking up and going to the bathroom and others referring to the process of opening the cabinet and taking medicine.

Rule 7 allows the system to identify the activity that the user is performing. The function *activityDescovery()* receives a set of identification codes referring to the sensors used during the activity, the set of appliances that the user used to perform it, and the local. It uses the resulting values of rules 8, 9 and 10 as parameters to perform this processing. The importance of 7 relays on the fact that the execution of a specific activity in a different hour or day than the usual may represent an unpredictable change of behavior, and the system should be aware of such change. In the case study, these rules would have the following structure.

2. activity.activityID = act001;
7. activity.identification = activityDescovery (activity.activityID, place.sensor_ID[], place.appliance[], local.room);
8. place.sensor_ID = door;
9. place.appliance = null;
10. local.room = bedroom;
2. activity.activityID = act002;
7. activity.identification = activityDescovery (activity.activityID, place.sensor_ID[], place.appliance[], local.room);
8. place.sensor_ID = medicineCabinet;
9. place.appliance = null;
10. local.room = bathroom;

In this example, the set of rules (2, 7, 8, 9 and 10) is also executed twice, once for each action that composes the activity (leaving the bedroom and taking the medicine). Rule 9 receives a *null* value because the user did not use any appliance to accomplish the activity. In a different scenario where the user watches TV or uses the stove to cook a meal, for instance, the rule would assume valid values.

The rules are used as the basis for the analysis of the level of accuracy of specific contexts. This means that they can contribute to the minimization of the problem of uncertainty. For instance, the system should be aware of the time spent by the user to finish the activities. Following, algorithm 1 describes how semantic rules could be used for context analysis.

The result produced by rule 1 should be analyzed against an established set of acceptable values, referring to the time. If the result is not satisfactory, it means that there are problems related to the activity. For instance, the value of *record.hour_begin* was not captured with precision, resulting in an uncertain time spent, or it may be related to problems with the user once he is not able to finish the task as usual. To solve this problem, all the processing of the rule *record.duration* should be executed again.

Algorithm 1 Example of a semantic rule for context analysis

```
1: procedure ACTIVITYANALYSIS(expectedDurationValues[])
2:     if getActivityDuration(parameters) not in expectedDurationValues[] then
3:         analyse(record.duration)
```

Despite the low complexity applied in this set of semantic rules, it is possible to state that, depending on the application, attribute grammars creates the possibility for the definition of much more complexes rules.

4 Uncertainty in context-aware systems

Situation refers to events that happen in an intelligent environment considering humans as main actors. The definition of a situation includes the user and every relevant data from entities that surround him. Identification of daily human activities can be used as examples of situations.

There are several problems related to the acquisition and processing of context data that may lead to uncertain situations. This problem impacts the decision making of context-aware systems once they use this data as the basis for reasoning. This section addresses essential features for a correct analysis of the quality of context information and how this be used for the identification of uncertainties.

4.1 Quality of Context

Data provided by physical sensors and resulted from processing in context-aware systems can vary according to its usefulness. Thus, the quality of the data provided by these sources to be used in orchestration of services is defined as Quality of Context (QoC) (Buchholz and Schiffers, 2003). According to (Buchholz and Schiffers, 2003), QoC is directly related to Quality of Services (QoS) and Quality of Devices (QoD). QoS strategies should be considered for the appropriate execution of applications, considering the goals of the system. Besides that, the data is gathered from sensors installed in the environment. Thus, the quality of these devices (QoD) may influence the provision of data for the system.

In (Buchholz and Schiffers, 2003) was defined a set of parameters to analyze the QoC. The first one refers to the *accuracy* of the data. The more accurate a piece of context information describes the real-world environment, the higher is its quality. This precision should be measurable in order to be computed. This is important because physical sensors are electronic devices that can fail and provide incorrect data. Thus, context-aware systems should be able

to analyze data aiming to verify its level of correctness by comparing it with pre-defined sets of expected values for each sensor. The accuracy of context data includes the veracity of past occurrences of it among the software agents. For instance, if a specific software agent frequently has to deal with imprecise data, the relevance of it must be analyzed carefully to avoid compromising reasoning and decision making.

Intelligent environments such as smart houses are generally characterized by their dynamicity. The *usefulness* of context data can change rapidly, i.e., one specific context data can be static for an extended period, and another can become outdated in a matter of seconds. Thus, the system must use *up-to-date* information whenever possible to ensure the proper decision making.

QoC can also be classified through the definition of another set of parameters (Hoyos et al., 2016). The type of *acquisition* is essential to analyze the source of data. This includes not only physical sensors but also the result of processing data. Thus, it is possible to assume that the software can also be seen as a source of data. Another parameter to be considered is the *types of representation* of context data. This is related to aspects like units, formats, and ranges that a set of data will have to ensure that the system will correctly interpret it. At last, context data should be analyzed regarding its *importance* for the software agents, i.e., how useful and relevant it is considering the goals of the system.

The approaches provided by (Buchholz and Schiffers, 2003) and (Hoyos et al., 2016) complement each other and have a substantial contribution to ensure that sensor data is consistent, complete, and relevant and, thus, can be used by context-aware systems. The following tuple formalizes it:

$$\text{QoC} = \langle A, C, Acq, R, I \rangle$$

Where:

- A refers to the accuracy of the data;
- C represents the set of acceptable values for one specific situation;
- Acq is the source of the data;
- R refers to the format of the data;
- I stores a value that ranks its usefulness.

QoC of a piece of data is ensured after the validation of each element and can be represented as an intersection of them, as follows:

$$\text{QoC} = \text{A} \cap \text{C} \cap \text{Acq} \cap \text{R} \cap \text{I}$$

These parameters are essential when dealing with context-aware systems. Any problem related to them may affect the orchestration of services and applications. If the system uses low-quality data, the user experience may be affected drastically (Buchholz and Schiffers, 2003).

The quality of context information is directly related to the goals of the system, i.e., depending on the type of adaptation that the system may assume. The same set of data may or may not be relevant (Hoyos et al., 2016). The level of quality of a situation is restricted to the quality of the set of context data. Different software agents from the same system may use the same set of data to build different scenarios (Buchholz and Schiffers, 2003).

Considering the scope of this paper, scenarios refer to the system's understanding of situations. Context-aware systems must be able to build scenarios with a high level of accuracy, considering the situations of the environment.

The QoC influences the analysis of each of these scenarios to represent the real-world situation with more precision. Besides that, the dissemination of useful QoC data prevents the system from using outdated information and minimizes problems caused by low-QoC data. Figure 3 presents a generic scheme of the dissemination of context data after analysis if its quality.

After receiving data from the environment, the system performs an analysis to ensure its quality. This includes all the features described above. Then, the sets of data are sent to their respective software agents, where they are properly used and can be shared with others, whenever requested. While waiting for the response of requesting information from another agent, the former may use speculative computation belief values to fill in the required information and not interrupt processing (Satoh, 2005), (Oliveira, 2017) — more details on this approach in (Freitas et al., 2019).

Uncertain scenarios are created from the use of low quality of context data. This is one of the main obstacles to overcome for the improvement of context-aware systems. The following section presents a proposal to tackle this problem, where the main goal is to identify elements that cause uncertainty.

4.2 Identification of uncertain situations

Context-aware systems to assist humans in ADL environments frequently may face imperfect information. Considering the scope of this paper, we characterize imperfect information as any data provided by physical sensors that do not reflect what happens in the environment with a satisfactory level of accuracy. This includes outdated, wrong, or missing data. The concept of accuracy may change according to the goals of the system. Besides that, assuming that systems can be seen as a source of information, after processing raw data, they also can generate imperfect information.

The uncertainty generated by incomplete context data may compromise the correct interpretation of real-world situations. Even considering a system with consistent and well-defined goals, if the data used as input does not have good quality, the result of processing will not have the most appropriated output.

The minimization of negative impacts caused by low-quality context data must be addressed. The approach presented in this paper highlights the importance of the identification of uncertain situations.

According to (Panayotov et al., 2018), uncertainty can be identified through the analysis of some features present in the context. Domains that involve monitoring human activities can be highly dynamic. Inputs used by context-aware systems may present small differences even when they represent the same context. Such systems should be flexible enough to calculate this and understand these discrepancies. This allows the system to create different scenarios that possibly represent the same situation. Thus, it is possible to create ranks considering the accuracy of the sensed situation regarding the real world. The process of identifying the sources of uncertainty may have high computation costs and time-consuming. The analysis of incom-

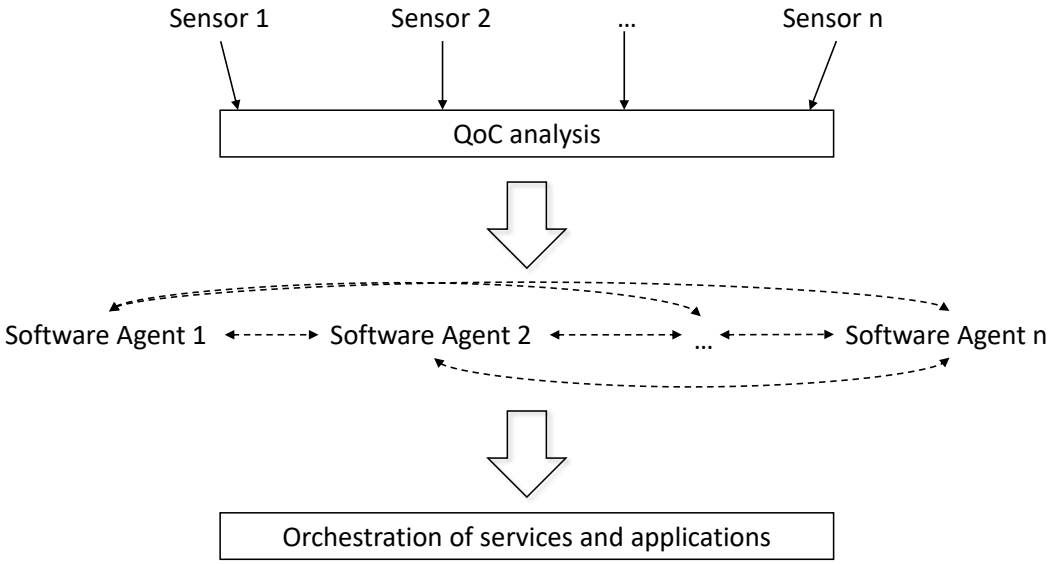

Figure 3 Context data sharing among software agents.

plete data must be performed for each entity that composes a situation.

According to (Uusitalo et al., 2015) and (Regan et al., 2002), uncertainty has several sources and can be categorized. Each of the types can not be isolated analyzed, especially in dynamic environments. It is natural for context-aware systems to deal with a certain level of **randomness**. In other words, it is not possible to precise with one hundred percent accuracy what happens in a situation even though probabilistic models have been used to improve these outputs. Besides that, problems related to the well functioning of sensors also must be tackled. These electronic devices can generate wrong data when not well-calibrated, for instance. Thus, context-aware systems must validate environmental data before using it as input for reasoning. Wrong **measurements** of the weight of users, the temperature of the environment, or the exact position of them are amongst the standard errors. This type of error affects the sample data from the environment that the system uses to build situations (**systematic**). Consequently, it will not reflect the real-world and will affect the orchestration of services. Still, according to (Uusitalo et al., 2015), it is difficult to find errors related to context data after the definition of situations based on inaccurate measurements.

The natural **dynamicity** of context data affects its usefulness for the interpretation of situations. This is another problem that may generate uncertainty once the system will have to use obsolete data as input for decision making. The definition of ranges of acceptable values front the same scenario is seen as an excellent approach to tackle this problem.

Another source of uncertainty is related to the **modelling**, i.e., creating systems capable of understanding what happens in intelligent environments. Many times it is impossible to evidence all necessary variables that involve the entities in such dynamic and productive environments. This problem affects the system in the sense that, if it is not well-defined, the range of services and applications will be fewer than it could. In other words, the system will not be as aware of the context as it could.

These types of uncertainty can also be categorized in a broader classification: **aleatoric** and **epistemic** (Helton, 1997), (Senge et al., 2014). The former includes any problems related to hardware failures or problems of communication. This may affect the behavior of the system, making it act unexpectedly. **Randomness** and **Measurements** categories can be associated with this type of uncertainty. Epistemic uncertainty is related to problems of interpretation of the context, i.e., the system does not have enough knowledge resources to resolve what happens in the environment. Usually, this is affected by aleatoric uncertainty. It also may be a result of modeling problems. **Systematic** and **Modelling** categories can be associated to this type.

5 Proposal

Previous sections described in detail different approaches to validate the context data. They can be applied separately according to the goals of the system. Also, they can complement themselves, being used in different steps of a more comprehensive analysis, aiming to improve the accuracy of environment's interpretation. Following, figure 4 presents an approach that encompasses the refinement of context data through attribute grammar, QoC analysis, and uncertainty identification. The main objective is to ensure that the system uses only data that successfully passed through all the modules before decision makings.

Six modules compose this framework, including the input, output and knowledge base. The validation of the sensor data is performed sequentially in three modules. First, its structure is verified through the attribute grammar. After that, the quality of context is analyzed, considering the parameters formalized in section 4.1. Before generating an

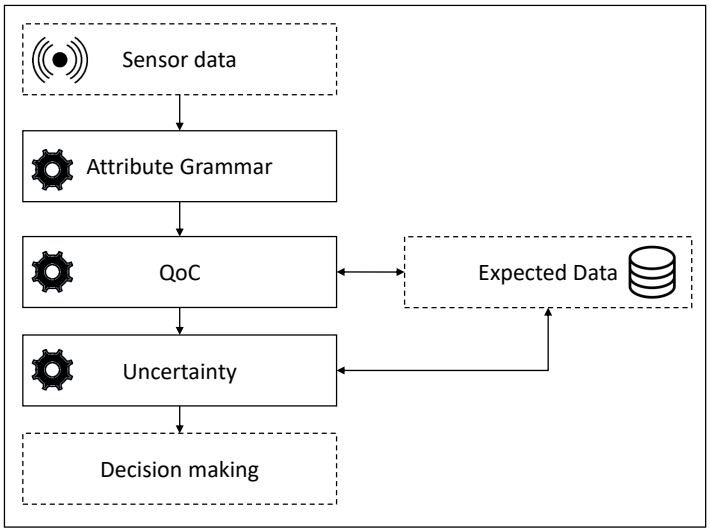

Figure 4 Context data refinement for decision making.

output, the last verification refers to uncertainty identification, where the system searches for imperfect data that may compromise the correct functioning of the system and impact the decision making.

The following sections describe the modules responsible for validating the data provided by sensors.

5.1 Attribute grammar verification module

This module is responsible for receiving data gathered from the environment and performs the first verification. It refers to the validation of its structure, including syntax and semantics. The set of data is applied to the grammar's production rules, where they will be analyzed.

The symbols in the grammar were defined following a strict and formal structure. Thus, context data should obey the restrictions that compose their axiom. After receiving the set of data, an AST is generated. This tree will be used to analyze the input values. After completing this process, the system is capable of verifying if the set of data represents a situation in a coherent way. This is defined by analyzing synthesized and inherited attributes for the values within the nodes of the AST.

The last data validation of this module refers to the execution of inference rules. The definition of which ones will be executed depends on the type of context data. The main goal of this phase is to generate knowledge to complement the data acquired from the environment. For instance, to perform the calculation of an activity's duration or to determine the exact location of the user based on raw data that refers to coordinates.

The importance of this is in the fact that if the system can generate more knowledge based on the AG's values, this could be used to minimize problems in the other modules (QoC and Uncertainty). Besides that, it allows the system to verify if the sensors are sending adequate data and if the user is following patterns of behavior. Any significant changes related to this may configure problems with the system or with the user, and the responsible entities should be notified. If any of these scenarios is confirmed, the analysis of the context data is interrupted, and the other two modules are not executed until the problem is solved.

However, if the validation of this module is completed, it means that the context data is well structured. Then the next phase is to send it to the QoC verification module.

5.2 QoC verification module

After ensuring the structure of the data that composes a situation, it is crucial to analyze its level of quality. Sensors that do not work well may provide imperfect data, and the processing of such data generates new data that is also imperfect. Thus, it is essential to verify if the system is using relevant data for the orchestration of services.

It is essential to differentiate the verification provided by AGs and the one performed in this module. To illustrate the difference between them, an example of a bathroom scale that sends the random user's weight to the system. The data is well-structured by considering past measurements it indicates a defective value.

There are several parameters that can be taken into consideration to ensure the quality of context data. New parameters can be incorporated and/ or existing ones might be ignored, depending on the situation and the system's goals.

Considering the domain of Activity Daily Living, the parameters *accuracy, usefulness, up-to-dateness, types of acquisition and representation* and *importance* are considered as being essential to ensure high quality of the context data. Each of them is compared to a knowledge base. This base stores previously validated data from past situations. Also, it stores acceptable ranges of values for each type of data. If any of the parameters is not successfully validated, the system should take some safety measures to avoid using imperfect data. Requisition of new data from sensors, starting an interaction with the user, sending notifications to service providers, or just ignoring a specific piece of data

can be some of the precautions taken by the system.

The process of validating the parameters does not ensure that the data has good quality. First, each parameter is verified individually. Then, they must be analyzed together, and common elements in their structure must be identified. Only data with all these characteristics are considered to have high QoC.

The process of analyzing the QoC of a piece of data can be very complicated, depending on the number of software agents and their goals. An out-dated data may be irrelevant for one agent but very accurate for another. These agents may have different functions, making some of them to be more sensitive to context changes than others. Thus, before discarding a piece of data, the system must verify, in this module, its usefulness to all software agents.

A context data is considered to have high quality if it passes through all these verification steps.

5.3 Uncertainty verification module

The main goal of this module is to identify problems related to context data that were not previously addressed. From this identification, the system can minimize negative impacts on the user by assuming safety measures to ensure its most appropriate behavior.

The first two modules are responsible for different types of validation of the context data and complement each other. Despite the data having a well-structure (AG verification) and high quality (QoC verification), sometimes it might not be as complete as it could. Thus, this module performs additional verification.

Due to the natural dynamicity of ADL environments, context data may become obsolete rapidly, as described in section 4.1. Besides that, the random behavior of sensors makes them send wrong measurements to the system, and this data can not be used for decision making.

Considering these issues, this module must create frequent communication with the knowledge base to verify if the current context data is as complete and useful as it should. Among other information, the knowledge base stores data from situations (activities) that were evidenced in the environment in the past. The system performs queries to verify if the context data is reliable. This process prevents the use of random data during the orchestration of services.

Besides that, if the system identifies that it needs specific context data that was not collected by the sensors to build a situation of context, it can use data from the registers of past situations. This allows the system to complete the process of decision making. However, users should be notified about the use of old data.

Aiming to validate this proposal, the next section presents the description of each of these steps in detail.

6 Validation of situation data

This section aims to present a formal representation of situations of context and, from this, describe how uncertain aspects of it can be identified for further analysis.

Intelligent environments analyze the current states of entities, identifying the relations among them to create a computational abstraction that represents real-world contexts. These real-world contexts will be called situations. They refer to specific settings of the environment, in specific periods of time, and involving specifics sets of entities.

The set of values from situations must be validated according to the restrictions of the module of the framework described in section 5. Considering that one situation is composed of a set of context values received from sensors, it can be formally represented by the following tuple:

$$S_i = \langle v_1, v_2, ..., v_k, ..., v_n \rangle$$

Where:

- i refers to a specific situation of context;
- n is the total number of context data used to compose the situation S_i;
- $1 < k < n$.

Each of the values v represents context data provided by physical sensors or the result of any processing that involved data from them. In this case, the context-aware system itself is also seen as a source of information. The situation S is characterized by the set of values (context data). It may have as many values as possible. The larger is the amount of context data in its tuple; the more accurate will the representation of the environment and, consequently, the more accurate will be the actions that can be performed. The absence of one or more of them may compromise the resulting representation of the real-world context. For instance, if one of the missing values, v_k, refers to the current location of the user in a smart house, it might not be possible to recognize the activity he is performing, if this data is imperative for this inference.

One problem that may emerge regarding the data of the situation is that, for each of them, a set of expected values must be defined. Expected sets of values are stored in the knowledge base of the framework. Additional approaches can be applied for the election of the most suitable vales to fill the gaps of data. For instance, algorithms for predictions could use the existing data from the base to provide possible values with a satisfactory level of confidence. The specific prediction approach must consider the types of data and service goals. This allows the system to ensure that only valid states are being considered for the situation to be identified. For instance, it is common sense that the human body temperature can vary from 35º Celsius to 41º (in case of fever). If v_k is equal to 31º Celsius and refers to the body temperature of the user in an Ambient Assisted Living, the system should identify that this characterizes a health problem with the user or it is not a valid number and, at least, not consider it for the analysis of the situation.

The formal specification of situations allows context-aware systems to take proper actions. They are functions of the situation S_i and affect the current state of entities, such as users and devices, as well as the environment itself. Generically, actions produce results and can be represented as follows:

$$\begin{aligned} action_1(S_1) &\longrightarrow result \\ action_2(S_1) &\longrightarrow result \\ &... \\ action_n(S_1) &\longrightarrow result \end{aligned}$$

In context-aware systems, actions refer to decision making based on environmental data. This includes the execution of applications or services, emission of alerts or warnings, and adaptation of appliances. The result of an action represents the consequence of its execution, i.e., how the action affects the environment. One situation S can be used as the basis for of execution of one or more actions.

The context values that compose situations are validated in all the modules of the framework. Following, algorithm 2 describes how the verification is performed in the *Uncertainty verification module* and its interactions with the knowledge base in order to prepare the situation to be used for the system's adaptation and decision making.

Algorithm 2 Analysis of the set of values of situation S

```
1:  procedure SITUATIONDATAANALYSIS(S[], expectedValues[])
2:      j ← 0
3:      for i ← 0 to length(S[]) do
4:          occurences[i] ← False
5:          while j < length(expectedValues[]) OR occurences[i] is False do
6:              if S[i] == expectedValues[j] then
7:                  occurences[i] ← True
8:              j ← j + 1
9:          j ← 0
10:     if CheckExistenceFalseValues(occurences[]) then
11:         runSensorDiagnosis(occurences[], S[])
12:         NotS[] ← getFalseValues(occurences[], S[])
13:         ValidS[] ← EliminateInvalid(S[], NotS[])
14:         r ← runActions(ValidS[])
15:         if !r then
16:             messageAlert()
17:             interaction(S[], occurences[])
```

The procedure receives the set of context data that composes the situation S and an array with values that are expected for that situation. The array with the expected values is a result of queries to the knowledge base. An auxiliary array (*occurrences[]*) is created with *False* values, in lines 3 and 4. After the verification, their states are changed to *True* if the corresponding sensor data has a value that is within the pre-defined range. In lines 5 and 6, the elements of $S[]$ are compared to the set of expected values for that specific situation. If this comparison results in a match, the state of that element, in *occurances*[], is changed to *True* (line 7). This routine will be finished after executing these steps for all sensor data.

Line 10 verifies the existence of invalid context data. If the return of the function *CheckExistenceFalseValues()* is *True*, it means that the situation S has null values or out of the pre-defined set of acceptable range (*False* state). In this case, a set of actions must be executed. First, the system starts diagnosis tests for the sensor(s) responsible(s) for sending the invalid context data (line 11). The function *runSensorDiagnosis* analyses hardware failures, checking if the sensor is operating, and communication problems, starting requisitions and verifying the received data. If problems are detected, the system sends message alerts to the responsible entities. After that, in the function *getFalseValues* (line 12), the system creates an array with invalid data. The array will be used to remove invalid data from the composition of S. The following tuples formalize this situation after the execution of this line code.

$$S_i = \langle v_1, v_2, ..., v_n \rangle$$
$$NotS_i = \langle v_1, v_2, ..., v_m \rangle$$

Where:

- $NotS_i$ contains only invalid data for the situation S_i;
- $NotS_i \notin S_i$;
- $m \leq n$.

A new array with only valid context data is created with the calling for the *EliminateInvalid* function (line 13). This process is executed because, even with missing values, there may be actions that could be performed for the situation S (line 14). If not possible, an appropriate message is sent to the user (lines 15 and 16). The function *messageAlert* identifies the invalid values and, depending on the type and amount, it takes different decisions, such as alert the user about the missing data or request new information to fill the gaps. The instructions of lines 14, 15, and 16 refer to actions from the *decision making* module. Based on the validation of context data from the three *verification* modules, the system analyzes which actions could be triggered to adapt the environment. Message alerts and interactions with the user may be necessary to improve the orchestration of services.

Intelligent environments are related to the paradigm of ubiquitous computing where a systems work autonomously, with minimal explicit interaction with users (Weiser, 1993). However, according to (Lim and Dey, 2011), the presentation of uncertain situations for them to solve helps the system to improve its knowledge. Consequently, the assistance provided by it will be more accurate and personalized. Considering this, in line 17, the specific requisition of context data through interaction with the user is performed, aiming to improve the QoC and, with this, minimize the impact that incomplete context data can have in the decision making.

7 Discussion and final considerations

Context-aware systems frequently face problems related to the necessary data to be used as input for decision making. Different causes may impact the orchestration of services and applications. These phenomena can compromise the correct adaptation of the system front specific situations.

Smart houses are seen as fertile research filed for the development of context-aware systems. Solutions for Ambient Assisted Living and Activity Daily Living aim to provide automation of tasks, creating a high level of assistance for humans.

The main contribution of this paper was to tackle the problem of uncertainty in daily human activities through the analysis of a sequence of essential issues to ensures the correct structure of the data, high Quality of Context (QoC)

data, and the identification of the sources of uncertain situations.

First, a new approach to validate context data was presented. Advantages of using attribute grammars as the formalism to ensure the structure of the data was described. Semantic and evaluation rules were applied to a case study to validate the proposal. After that, the importance of using only high QoC as input for decision making was discussed. Several characteristics were analyzed, and the formalization of QoC was presented. The approach was validated through the description of the importance of sharing only high QoC among software agents of the system.

One useful approach for dealing with uncertain situations is through the identification of the source of it. Thus, after analyzing the previous issues, it was possible to tackle this problem. It was presented a formalization of situations of context aiming to facilitate the interpretation of its content by the system. An algorithm was described with a set of functions to be executed when the system identifies problems with the input data. They represent part of the decision making.

According to (White et al., 2018), the definition of uncertainty models helps the identification of its source and the definition of optimal solutions for decision making. They can generate unique outputs based on the processing of imprecise data. This can be applied to context-aware situations formalized through an attribute grammar approach when the data from sensors are not enough to define a situation.

It is believed that attribute grammars can be allied to other fields of artificial intelligence. Thus, the next step of the project is to merge them, aiming for the improvement of solutions for context-aware systems.

Acknowledgment

"This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020".

REFERENCES

Anderson, M. P., Woessner, W. W., and Hunt, R. J. (2015). Chapter 10 - forecasting and uncertainty analysis. In Anderson, M. P., Woessner, W. W., and Hunt, R. J., editors, *Applied Groundwater Modeling*, pages 443–491. Academic Press, San Diego, 2nd ed. edition.

Bolton, M. L., Siminiceanu, R. I., and Bass, E. J. (2011). A systematic approach to model checking human–automation interaction using task analytic models. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 41(5):961–976.

Buchholz, T. and Schiffers, M. (2003). Quality of context: What it is and why we need it. In *In Proceedings of the 10th Workshop of the OpenView University Association: OVUA'03*.

Bürger, C., Karol, S., and Wende, C. (2010). Applying attribute grammars for metamodel semantics. *ECOOP 2010 Workshop Proceedings - International Workshop on Formalization of Modeling Languages, FML'10*.

D'Aniello, G., Loia, V., and Orciuoli, F. (2017). Adaptive goal selection for improving situation awareness: the fleet management case study. *Procedia Computer Science*, 109:529–536.

Freitas, L., Henriques, P. R., and Novais, P. (2018). Uncertainty in context-aware systems: A case study for intelligent environments. In Álvaro Rocha, Adeli, H., Reis, L. P., and Costanzo, S., editors, *Trends and Advances in Information Systems and Technologies, WorldCist2018*, volume 745 of *Advances in Intelligent Systems and Computing*, pages 225–231. Springer International Publishing, 1 edition.

Freitas, L. O., Henriques, P. R., and Novais, P. (2019). Context-awareness and uncertainty: Current scenario and challenges for the future. In Novais, P., Jung, J. J., Villarrubia González, G., Fernández-Caballero, A., Navarro, E., González, P., Carneiro, D., Pinto, A., Campbell, A. T., and Durães, D., editors, *Ambient Intelligence – Software and Applications –, 9th International Symposium on Ambient Intelligence*, pages 174–181, Cham. Springer International Publishing.

Giese, M., Mistrzyk, T., Pfau, A., Szwillus, G., and von Detten, M. (2008). Amboss: A task modeling approach for safety-critical systems. In Forbrig, P. and Paternò, F., editors, *Engineering Interactive Systems*, Berlin. Springer Berlin Heidelberg.

Helton, J. (1997). Uncertainty and sensitivity analysis in the presence of stochastic and subjective uncertainty. *Journal of Statistical Computation and Simulation*, 57(1-4):3–76.

Hoyos, J. R., Garca-Molina, J., Bota, J. A., and Preuveneers, D. (2016). A model-driven approach for quality of context in pervasive systems. *Comput. Electr. Eng.*, 55(C):39–58.

Knuth, D. E. (1968). Semantics of context-free languages. In *In Mathematical Systems Theory*, pages 127–145.

Lim, B. Y. and Dey, A. K. (2011). Investigating intelligibility for uncertain context-aware applications. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, UbiComp '11, pages 415–424, New York, NY, USA. ACM.

Lyons, P., Cong, A., Steinhauer, H., Marsland, S., Dietrich, J., and Guesgen, H. (2010). Exploring the responsibilities of single-inhabitant smart homes with use cases. *JAISE*, 2:211–232.

Mori, G., Paterno, F., and Santoro, C. (2002). Ctte: support for developing and analyzing task models for interactive system design. *IEEE Transactions on Software Engineering*, 28(8):797–813.

Noor, M. H. M., Salcic, Z., and Wang, K. I.-K. (2016). Enhancing ontological reasoning with uncertainty handling for activity recognition. *Know.-Based Syst.*, 114(C):47–60.

Oliveira, T. J. M. (2017). *Clinical Decision Support: Knowledge Representation and Uncertainty Management*. PhD thesis, University of Minho - Doctoral Programme of Biomedical Engineering, R. da Universidade, 4710-057, Braga - Campus de Gualtar.

Panayotov, D., Grief, A., Merrill, B. J., Humrickhouse, P. W., Murgatroyd, J. T., Owen, S., and Saunders, C. (2018). Uncertainties identification and initial evaluation in the accident analyses of fusion breeder blankets. *Fusion Engineering and Design*, 136:993 – 999. Special Issue: Proceedings of the 13th International Symposium on Fusion Nuclear Technology (ISFNT-13).

Regan, H. M., Colyvan, M., and Burgman, M. A. (2002). A taxonomy and treatment of uncertainty for ecology and conservation biology. *Ecological Applications*, 12(2):618–628.

Satoh, K. (2005). Speculative computation and abduction for an autonomous agent*. *IEICE - Trans. Inf. Syst.*, E88-D(9):2031–2038.

Senge, R., Bösner, S., Dembczynski, K., Haasenritter, J., Hirsch, O., Donner-Banzhoff, N., and Hüllermeier, E. (2014). Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty. *Inf. Sci.*, 255:16–29.

Uusitalo, L., Lehikoinen, A., Helle, I., and Myrberg, K. (2015). An overview of methods to evaluate uncertainty of deterministic models in decision support. *Environ. Model. Softw.*, 63(C):24–31.

Weiser, M. (1993). Ubiquitous computing. *Computer*, 26(10):71–72.

White, J. T., Fienen, M. N., Barlow, P. M., and Welter, D. E. (2018). A tool for efficient, model-independent management optimization under uncertainty. *Environ. Model. Softw.*, 100:213 – 221.